UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/713,024	11/17/2003	Allen N. L. Lau	13527-1	9733

1059 7590 02/26/2007

BERESKIN AND PARR
40 KING STREET WEST
BOX 401
TORONTO, ON M5H 3Y2
CANADA

EXAMINER	
RAMPURIA, SATISH	
ART UNIT	**PAPER NUMBER**
2191	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/26/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/713,024	LAU ET AL.	
	Examiner	Art Unit	
	Satish S. Rampuria	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 November 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *17 November 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *3/24/05,2/19/04*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the application filed on November 17, 2003.

2. Claims 1-20 are pending.

Information Disclosure Statement

3. An initialed and dated copy of Applicant's IDS form 1449 filed on 3/24/05 and 2/19/04 are attached to the instant Office action.

Oath/Declaration

4. The Office acknowledges receipt of a properly signed oath/declaration filed November 17, 2003.

Specification

5. The disclosure is objected to because of the following informalities:

Appropriate correction is required.

6. The disclosure is objected to the use following acronyms. Applicant are required to provide the full form of the acronyms.

7. The use of the trademark/service mark "Java", "JVM", J2ME" has been noted through out the application. It should be appropriate or proper term (i.e., Java™) (see MPEP 608.01(v)) used, wherever it appears and be accompanied by the generic terminology (for details please visit http://www.sun.com/suntrademarks/index.html). Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

Drawings

8. The drawings were received on November 17, 2003. These drawings are acceptable by the examiner.

Claim Rejections - 35 USC § 112

9. The following is a quotation of the **second paragraph** of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

10. Claims 1-20 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Clarification and/or correction are required.

11. Claims 1, 13, 14, 17, 19, and 20 contain the trademark/trade name Java. Where a trademark or trade name is used in a claim as a limitation to identify or describe a particular material or product, the claim does not comply with the requirements of 35 U.S.C. 112, second paragraph. See Ex parte Simpson, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or trade name cannot be used properly to identify any particular material or product. A trademark or trade name is used to identify a source of goods, and not the goods themselves. Thus, a trademark or trade name does not identify or describe the goods associated with the trademark or trade name.

The rejection of the base claim is necessarily incorporated into the dependent claims.

Double Patenting

12. A rejection based on double patenting of the "same invention" type finds its support in the language of 35 U.S.C. 101 which states that "whoever invents or discovers any new and useful process ... may obtain a patent therefor ..." (Emphasis added). Thus, the term "same invention," in this context, means an invention drawn to identical subject matter. See *Miller v. Eagle Mfg. Co.*, 151 U.S. 186 (1894); *In re Ockert*, 245 F.2d 467, 114 USPQ 330 (CCPA 1957); and *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970).

A statutory type (35 U.S.C. 101) double patenting rejection can be overcome by canceling or amending the conflicting claims so they are no longer coextensive in scope. The filing of a terminal disclaimer cannot overcome a double patenting rejection based upon 35 U.S.C. 101.

13. Claims 1-20 provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 1-13,18-21, and 29-31 of copending Application No. 10/782,917 ***(hereinafter called '917)***. This is a provisional double patenting rejection since the conflicting claims have not in fact been patented.

Instant Claim	*'917 Claim*
1. A method of generating a target application from a reference application, the reference application being a Java application adapted to execute on a reference mobile device, the target application being configured for a target mobile device, the method comprising: a) unpacking the reference application into a plurality of class files; b) transforming the reference application into the target application by a plug-in, wherein the plug-in is adapted to transform a plurality of different reference applications into a corresponding plurality of target applications for a predetermined combination of the reference mobile device and the target mobile device.	1. A method of generating a target application from a reference application, the reference application being a Java application adapted to execute on a reference mobile device, the target application being configured for a target mobile device, the method comprising: a) unpacking the reference application into a plurality of class files; b) transforming the reference application into the target application by a plug-in, wherein the plug-in is adapted to transform a plurality of different reference applications into a corresponding plurality of target applications for a predetermined combination of the reference mobile device and the target mobile device.
2. The method of claim 1, wherein the reference application is in bytecode during the transformation step.	2. The method of claim 1, wherein the reference application is in bytecode during the transformation step.
3. The method of claim 2, wherein the plug-in	3. The method of claim 2, wherein the plug-in

comprises an instruction file and at least one library, wherein the transformation step comprises the instruction file instructing a transformation engine to modify a portion of the reference application with a selected software code stored in the library, wherein the portion of the reference application is not supported by the target mobile device.	comprises an instruction file and at least one library, wherein the transformation step comprises the instruction file instructing a transformation engine to modify a portion of the reference application with a selected software code stored in the library, wherein the portion of the reference application is not supported by the target mobile device.
4. The method of claim 3, wherein the transforming step comprises modifying at least one of a plurality of class files in the reference application.	4. The method of claim 3, wherein the transforming step comprises modifying at least one of a plurality of class files in the reference application.
5. The method of claim 4, wherein the transformation step comprises adding a new class file to the reference application.	5. The method of claim 4, wherein the transformation step comprises adding a new class file to the reference application.
6. The method of claim 4, wherein the modifying step comprises at least one action selected from the group of: adding a new method, renaming an existing method,	6. The method of claim 4, wherein the modifying step comprises at least one action selected from the group of: adding a new method, renaming an existing method,

replacing a first object method call with a second object method call, replacing the first object method call with a static method call, renaming a constant pool entry, and inserting a new inner class to an existing class.	replacing a first object method call with a second object method call, replacing the first object method call with a static method call, renaming a constant pool entry, and inserting a new inner class to an existing class.
7. The method of claim 6, further comprising saving the target application to a computer readable medium.	7. The method of claim 6, further comprising saving the target application to a computer readable medium.
8. The method of claim 7, further comprising repeating step (a) and step (b) to transform the plurality of different reference applications into the plurality of corresponding target applications.	8. The method of claim 7, further comprising repeating step (a) and step (b) to transform the plurality of different reference applications into the plurality of corresponding target applications.
9. The method of claim 1, further comprising selecting a predetermined plug-in from a plurality of the plug-ins, the predetermined plug-in corresponding to the predetermined combination of the reference mobile device and the target mobile device, each of the	9. The method of claim 1, further comprising selecting a predetermined plug-in from a plurality of the plug-ins, the predetermined plug-in corresponding to the predetermined combination of the reference mobile device and the target mobile device, each of the

plurality of the plug-ins corresponding to a different combination of the reference mobile device and the target mobile device.	plurality of the plug-ins corresponding to a different combination of the reference mobile device and the target mobile device.
10. The method of claim 3, further comprising repackaging the target application into executable code.	10. The method of claim 3, further comprising repackaging the target application into executable code.
11. The method of claim 10, wherein the repackaging step further comprises obfuscating the target the class files of the target application.	11. The method of claim 10, wherein the repackaging step further comprises obfuscating the target the class files of the target application.
12. The method of claim 10, wherein the repackaging step further comprises pre-verifying the class files of the target application.	12. The method of claim 10, wherein the repackaging step further comprises pre-verifying the class files of the target application.
13. The method of claim 3, wherein the target application is a non-Java application.	13. The method of claim 3, wherein the target application is a non-Java application.

14. A system for transforming Java reference applications adapted to execute on a reference mobile device into corresponding target applications configured for a target mobile device, the system comprising: a) a transformation engine; and b) a plug-in comprising: i) an instruction file; and ii) a selected software code adapted to modify a portion of the reference application not supported by the target mobile device; wherein the transformation engine is adapted to access the instruction file, the instruction file being adapted to direct the transformation engine to identify the portion of the reference application and to modify the portion with the selected software code.	18. A system for transforming Java reference applications adapted to execute on a reference mobile device into corresponding target applications configured for a target mobile device, the system comprising: a) a transformation engine; and b) a plug-in comprising: i) an instruction file; and ii) a selected software code adapted to modify a portion of the reference application not supported by the target mobile device; wherein the transformation engine is adapted to access the instruction file, the instruction file being adapted to direct the transformation engine to identify the portion of the reference application and to modify the portion with the selected software code.
15. The system of claim 14, wherein the instruction file comprises an XML file.	19. The system of claim 18, wherein the instruction file comprises an XML file.
16. The system of claim 15, wherein the plug-in comprises a library, the library being	20. The system of claim 19, wherein the plug-in comprises a library, the library being

adapted to store a plurality of the software codes adapted to modify a plurality of the portions. 17. The system of claim 16, wherein the transformation engine is a Java application.	adapted to store a plurality of the software codes adapted to modify a plurality of the portions. 21. The system of claim 20, wherein the transformation engine is a Java application.
18. The system of claim 15, further comprising a plurality of the plug-ins, each of the plurality of the plug-ins corresponding to a different combination of reference and target mobile device, wherein the transformation engine is adapted to choose a selected one of the plug-ins corresponding to the different combination. 19. The system of claim 16, wherein the target applications are Java applications. 20. The system of claim 16, wherein the target applications are non-Java applications.	29. The system of claim 19, further comprising a plurality of the plug-ins, each of the plurality of the plug-ins corresponding to a different combination of reference and target mobile device, wherein the transformation engine is adapted to choose a selected one of the plug-ins corresponding to the different combination. 30. The system of claim 20, wherein the target applications are Java applications. 31. The system of claim 20, wherein the target applications are non-Java applications.

14. Claims 1-20 provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 1-13, 125-28, 36-38, and 39 of copending Application No. 10/975,346

(hereinafter called '346). This is a provisional double patenting rejection since the conflicting claims have not in fact been patented.

Instant Claim	*'346 Claim*
1. A method of generating a target application from a reference application, the reference application being a Java application adapted to execute on a reference mobile device, the target application being configured for a target mobile device, the method comprising: a) unpacking the reference application into a plurality of class files; b) transforming the reference application into the target application by a plug-in, wherein the plug-in is adapted to transform a plurality of different reference applications into a corresponding plurality of target applications for a predetermined combination of the reference mobile device and the target mobile device.	1. A method of generating a target application configured for a target mobile device from a Java reference application configured to execute on a reference mobile device, the method comprising: a) unpacking the reference application into a plurality of class files; b) transforming the reference application into the target application by a device plug-in, wherein the device plug-in is adapted to transform a plurality of different reference applications into a corresponding plurality of target applications for a predetermined combination of the reference mobile device and the target mobile device.
2. The method of claim 1, wherein the reference application is in bytecode during the transformation step.	2. The method of claim 1, wherein the reference application is in bytecode during the transformation step.
3. The method of claim 2, wherein the plug-in	3. The method of claim 2, wherein the device

comprises an instruction file and at least one library, wherein the transformation step comprises the instruction file instructing a transformation engine to modify a portion of the reference application with a selected software code stored in the library, wherein the portion of the reference application is not supported by the target mobile device.	plug-in comprises an instruction file and at least one library, wherein the transformation step comprises the instruction file instructing a transformation engine to modify a portion of the reference application with a selected software code stored in the library, wherein the portion of the reference application is not supported by the target mobile device.
4. The method of claim 3, wherein the transforming step comprises modifying at least one of a plurality of class files in the reference application.	4. The method of claim 3, wherein the transforming step comprises modifying at least one of a plurality of class files in the reference application.
5. The method of claim 4, wherein the transformation step comprises adding a new class file to the reference application.	5. The method of claim 4, wherein the transformation step comprises adding a new class file to the reference application.
6. The method of claim 4, wherein the modifying step comprises at least one action selected from the group of: adding a new method, renaming an existing method,	6. The method of claim 4, wherein the modifying step comprises at least one action selected from the group of: adding a new method, renaming an existing method,

replacing a first object method call with a second object method call, replacing the first object method call with a static method call, renaming a constant pool entry, and inserting a new inner class to an existing class.	replacing a first object method call with a second object method call, replacing the first object method call with a static method call, renaming a constant pool entry, and inserting a new inner class to an existing class.
7. The method of claim 6, further comprising saving the target application to a computer readable medium.	7. The method of claim 6, further comprising saving the target application to a computer readable medium.
8. The method of claim 7, further comprising repeating step (a) and step (b) to transform the plurality of different reference applications into the plurality of corresponding target applications.	8. The method of claim 7, further comprising repeating step (a) and step (b) to transform the plurality of different reference applications into the plurality of corresponding target applications.
9. The method of claim 1, further comprising selecting a predetermined plug-in from a plurality of the plug-ins, the predetermined plug-in corresponding to the predetermined combination of the reference mobile device and the target mobile device, each of the	9. The method of claim 1, further comprising selecting a predetermined device plug-in from a plurality of the device plug-ins, the predetermined device plug-in corresponding to the predetermined combination of the reference mobile device and the target mobile device,

plurality of the plug-ins corresponding to a different combination of the reference mobile device and the target mobile device.	each of the plurality of the device plug-ins corresponding to a different combination of the reference mobile device and the target mobile device.
10. The method of claim 3, further comprising repackaging the target application into executable code.	10. The method of claim 3, further comprising repackaging the target application into executable code.
11. The method of claim 10, wherein the repackaging step further comprises obfuscating the target the class files of the target application.	11. The method of claim 10, wherein the repackaging step further comprises obfuscating the class files of the target application.
12. The method of claim 10, wherein the repackaging step further comprises pre-verifying the class files of the target application.	12. The method of claim 10, wherein the repackaging step further comprises pre-verifying the class files of the target application.
13. The method of claim 3, wherein the target application is a non-Java application.	13. The method of claim 3, wherein the target application is a non-Java application.
14. A system for transforming Java reference	25. A system for transforming Java reference

applications adapted to execute on a reference mobile device into corresponding target applications configured for a target mobile device, the system comprising: a) a transformation engine; and b) a plug-in comprising: i) an instruction file; and ii) a selected software code adapted to modify a portion of the reference application not supported by the target mobile device; wherein the transformation engine is adapted to access the instruction file, the instruction file being adapted to direct the transformation engine to identify the portion of the reference application and to modify the portion with the selected software code.	applications adapted to execute on a reference mobile device into corresponding target applications configured for a target mobile device, the system comprising: a) a transformation engine; and b) a device plug-in comprising: i) an instruction file; and ii) a selected software code adapted to modify a portion of the reference application not supported by the target mobile device; wherein the transformation engine is adapted to access the instruction file, the instruction file being adapted to direct the transformation engine to identify the portion of the reference application and to modify the portion with the selected software code.
15. The system of claim 14, wherein the instruction file comprises an XML file.	26. The system of claim 25, wherein the instruction file comprises an XML file.
16. The system of claim 15, wherein the plug-in comprises a library, the library being adapted to store a plurality of the software	27. The system of claim 26, wherein the device plug-in comprises a library, the library being adapted to store a plurality of the software

codes adapted to modify a plurality of the portions.	codes adapted to modify a plurality of the portions.
17. The system of claim 16, wherein the transformation engine is a Java application.	28. The system of claim 27, wherein the transformation engine is a Java application.
18. The system of claim 15, further comprising a plurality of the plug-ins, each of the plurality of the plug-ins corresponding to a different combination of reference and target mobile device, wherein the transformation engine is adapted to choose a selected one of the plug-ins corresponding to the different combination.	36. The system of claim 26, further comprising a plurality of the device plug-ins, each of the plurality of the device plug-ins corresponding to a different combination of reference and target mobile device, wherein the transformation engine is adapted to choose a selected one of the device plug-ins corresponding to the different combination.
19. The system of claim 16, wherein the target applications are Java applications.	37. The system of claim 27, wherein the target applications are Java applications.
20. The system of claim 16, wherein the target applications are non-Java applications.	38. The system of claim 27, wherein the target applications are non-Java applications.
14. A system for transforming Java reference applications adapted to execute on a reference	39. A system for transforming a Java reference application configured to execute on a

mobile device into corresponding target applications configured for a target mobile device, the system comprising: a) a transformation engine; and b) a plug-in comprising: i) an instruction file; and ii) a selected software code adapted to modify a portion of the reference application not supported by the target mobile device; wherein the transformation engine is adapted to access the instruction file, the instruction file being adapted to direct the transformation engine to identify the portion of the reference application and to modify the portion with the selected software code. 18. The system of claim 15, further comprising a plurality of the plug-ins, each of the plurality	reference mobile device into a target application configured to execute on a target mobile device, the system comprising: a) a transformation engine; and b) a device plug-in adapted to transform the reference application into a target application, the device plug-in comprising: i) an instruction file; and ii) a selected software code adapted to modify a portion of the reference application not supported by the target mobile device; and c) an application plug-in adapted to instruct the transformation engine to modify at least a portion of the reference application; wherein the portion of the reference application does not execute optimally on the target mobile device after transformation of the reference application by the device plug-in;

of the plug-ins corresponding to a different combination of reference and target mobile device, wherein the transformation engine is adapted to choose a selected one of the plug-ins corresponding to the different combination.	wherein the transformation engine is adapted to access the instruction file, the instruction file being adapted to direct the transformation engine to identify the portion of the reference application and to modify the portion with the selected software code.

Claim Rejections - 35 USC § 102

15. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless --
(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

16. Claims 1-14 are rejected under 35 U.S.C. 102(e) as being anticipated by US Publication No. 2004/0015911 to Hinsley et al. (hereinafter, Hinsley).

Per claim 1:

Hinsley disclose:

1. A method of generating a target application from a reference application, the reference application being a Java application adapted to execute on a reference mobile device, the target application being configured for a target mobile device, the method comprising:

a) unpacking the reference application into a plurality of class files (paragraph [0028] "The translation from bytecode to Virtual Processor code may be undertaken on a central server, which may also provide verification of the Virtual Processor class files");

b) transforming the reference application into the target application by a plug-in (paragraph [0028] "Each client processor maintains its own native translator which it uses to translate the received Virtual Processor code into its own particular variety of native code"), wherein the plug-in is adapted to transform a plurality of different reference applications into a corresponding plurality of target applications (paragraph [0028] "each client device will translate into the appropriate native code for its own type of processor") for a predetermined combination of the reference mobile device and the target mobile device (paragraph [0030] "The present invention is expected to have particular application in the field of wireless communications, (wireless client networks), and specifically although not exclusively in the field of mobile cellular phone networks").

Per claim 2:

The rejection of claim 1 is incorporated and further, Hinsley disclose:

2. The method of claim 1, wherein the reference application is in bytecode during the transformation step (paragraph [0039] "FIG. 1, which has been described above, shows the way in which a JIT compiler within a JVM translates from processor-independent bytecode to

processor-dependent native code for running on a particular processor").

Per claim 3:

The rejection of claim 2 is incorporated and further, Hinsley disclose:

3. The method of claim 2, wherein the plug-in comprises an instruction file and at least one library (paragraph [0059] "A data tool...contains information about a class, including but not limited to the names, parameters and types of all constructors, fields, methods and other entities which make up the API of a class. A typical use for this would be for reflection (i.e. the functionality in java.lang.reflect in a Java Library)."), wherein the transformation step comprises the instruction file instructing a transformation engine to modify a portion of the reference application with a selected software code stored in the library, wherein the portion of the reference application is not supported by the target mobile device (paragraph [0039] "FIG. 1, which has been described above, shows the way in which a JIT compiler within a JVM translates from processor-independent bytecode to processor-dependent native code for running on a particular processor").

Per claim 4:

The rejection of claim 3 is incorporated and further, Hinsley disclose:

4. The method of claim 3, wherein the transforming step comprises modifying at least one of a plurality of class files in the reference application (paragraph [0042] "FIG. 2 illustrates in more detail the translation from class bytecode to native code...The class verifier if necessary loads additional classes to check the external references").

Per claim 5:

The rejection of claim 4 is incorporated and further, Hinsley disclose:

5. The method of claim 4, wherein the transformation step comprises adding a new class file to the reference application (paragraph [0042] "FIG. 2 illustrates in more detail the translation from class bytecode to native code...The class verifier if necessary loads additional classes to check the external references").

Per claim 6:

The rejection of claim 4 is incorporated and further, Hinsley disclose:

6. The method of claim 4, wherein the modifying step comprises at least one action selected from the group of: adding a new method (paragraph [0082] "executing the new method with the parameters"), renaming an existing method, replacing a first object method call with a second object method call (paragraph [0082] "converted to VP which loads all the parameters into VP registers before executing a gos (goto subroutine) instruction which has been fixed up to point to the destination method (this fixup may be statistically or dynamically bound"), replacing the first object method call with a static method call, renaming a constant pool entry, and inserting a new inner class to an existing class (paragraph [0042] "...The class verifier if necessary loads additional classes to check the external references...").

Per claim 7:

The rejection of claim 6 is incorporated and further, Hinsley disclose:

7. The method of claim 6, further comprising saving the target application to a computer readable medium (paragraph [0090] "The native translator 214 is quite a small piece of code (around 150 k, depending upon the processor), so that it can easily be stored in memory within an embedded system").

Per claim 8:

The rejection of claim 7 is incorporated and further, Hinsley disclose:

8. The method of claim 7, further comprising repeating step (a) and step (b) to transform the plurality of different reference applications into the plurality of corresponding target applications (paragraph [0028] "each client device will translate into the appropriate native code for its own type of processor").

Per claim 9:

The rejection of claim 1 is incorporated and further, Hinsley disclose:

9. The method of claim 1, further comprising selecting a predetermined plug-in from a plurality of the plug-ins, the predetermined plug-in corresponding to the predetermined combination of the reference mobile device and the target mobile device (paragraph [0049] "FIG. 2, further details will be given of the two-stage translation from class bytecode 210 into native code 230. As previously described, the class verifier 211 checks the class bytecode for validity. The class verifier may in some embodiments be incorporated within the jcode translator, in which case the class bytecode 210 is passed straight to the jcode translator 212 as shown by the arrow 240"), each of the plurality of the plug-ins corresponding to a different combination of the reference

mobile device and the target mobile device (paragraph [0045] "The use of the preferred embodiment within heterogeneous multiprocessor environment is shown schematically in FIG. 4. This should be compared with the corresponding prior art approach shown in FIG. 3").

Per claim 10:

The rejection of claim 3 is incorporated and further, Hinsley disclose:

10. The method of claim 3, further comprising repackaging the target application into executable code (paragraph [0091] "Both the jcode translator and the native translator are themselves preferably written in VP code and can thus be translated (using the native translator itself) to run on any desired platform").

Per claim 11:

The rejection of claim 10 is incorporated and further, Hinsley disclose:

11. The method of claim 10, wherein the repackaging step further comprises obfuscating the target the class files of the target application (paragraph [0028] "each client device will translate into the appropriate native code for its own type of processor").

Per claim 12:

The rejection of claim 10 is incorporated and further, Hinsley disclose:

12. The method of claim 10, wherein the repackaging step further comprises pre-verifying the class files of the target application (paragraph [0042] "FIG. 2 illustrates in more detail the translation from class bytecode to native code...The class verifier if necessary loads additional

classes to check the external references").

Per claim 13:

The rejection of claim 3 is incorporated and further, Hinsley disclose:

13. The method of claim 3, wherein the target application is a non-Java application (paragraph [0087] "the debug information is converted to the same format used for non-Java parts of the system").

Per claim 14:

Hinsley disclose:

14. A system for transforming Java reference applications adapted to execute on a reference mobile device into corresponding target applications configured for a target mobile device, the system comprising:

a) a transformation engine (paragraph [0023] "the program bytecode into machine independent virtual processor code which uses an instruction set of a virtual processor"); and

b) a plug-in comprising:

i) an instruction file (paragraph [0053] "A java class file"); and

ii) a selected software code adapted to modify a portion of the reference application not supported by the target mobile device (paragraph [0039] "FIG. 1, which has been described above, shows the way in which a JIT compiler within a JVM translates from processor-independent bytecode to processor-dependent native code for running on a particular processor");

wherein the transformation engine is adapted to access the instruction file, the instruction file being adapted to direct the transformation engine to identify the portion of the reference application and to modify the portion with the selected software code (paragraph [0059] "A data tool...contains information about a class, including but not limited to the names, parameters and types of all constructors, fields, methods and other entities which make up the API of a class. A typical use for this would be for reflection (i.e. the functionality in java.lang.reflect in a Java Library).").

Claim Rejections - 35 USC § 103

17. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

18. Claims 15-20 rejected under 35 U.S.C. 103(a) as being unpatentable over Hinsley in view of US Publication No. 2003/0056207 to Fischer et al. (hereinafter, Fischer).

Per claim 15:

The rejection of claim 14 is incorporated and further, Hinsley does not explicitly disclose wherein the instruction file comprises an XML file.

However, Fischer discloses in an analogous computer system wherein the instruction file comprises an XML file (paragraph [0045] "Library service...contains an open catalog /file/ interface and parser for XML parsing...storage encryptor").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of wherein the instruction file comprises an XML file as taught by Fischer in the translating computer programs into native code for wirless communication device as taught by Hinsley. The modification would be obvious because of one of ordinary skill in the art would be motivated to have an instruction file as an XML file to provide installation and deployment of a platform independent mobile framework and offline applications on a mobile devices (paragraph [0010] and [0048]).

Per claim 16:

The rejection of claim 15 is incorporated and further, Hinsley disclose:

16. The system of claim 15, wherein the plug-in comprises a library, the library being adapted to store a plurality of the software codes adapted to modify a plurality of the portions (paragraph [0059] "A data tool...contains information about a class, including but not limited to the names, parameters and types of all constructors, fields, methods and other entities which make up the API of a class. A typical use for this would be for reflection (i.e. the functionality in java.lang.reflect in a Java Library).").

Per claim 17:

The rejection of claim 16 is incorporated and further, Hinsley disclose:

17. The system of claim 16, wherein the transformation engine is a Java application (paragraph [0039] "FIG. 1, which has been described above, shows the way in which a JIT compiler within a JVM translates from processor-independent bytecode to processor-dependent native code for

running on a particular processor").

Per claim 18:

The rejection of claim 15 is incorporated and further, Hinsley disclose:

18. The system of claim 15, further comprising a plurality of the plug-ins, each of the plurality of the plug-ins corresponding to a different combination of reference and target mobile device, wherein the transformation engine is adapted to choose a selected one of the plug-ins corresponding to the different combination (paragraph [0039] "FIG. 1, which has been described above, shows the way in which a JIT compiler within a JVM translates from processor-independent bytecode to processor-dependent native code for running on a particular processor").

Per claim 19:

The rejection of claim 16 is incorporated and further, Hinsley disclose:

19. The system of claim 16, wherein the target applications are Java applications (paragraph [0087] "converted to a format suitable for the environment in which the JVM is running").

Per claim 20:

The rejection of claim 16 is incorporated and further, Hinsley disclose:

20. The system of claim 16, wherein the target applications are non-Java applications (paragraph [0087] "the debug information is converted to the same format used for non-Java parts of the system").

Conclusion

19. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to **Satish S. Rampuria** whose telephone number is **(571) 272-3732**. The examiner can normally be reached on **8:30 am to 5:00 pm** Monday to Friday except every other Friday and Wednesday and federal holidays. Any inquiry of a general nature or relating to the status of this application should be directed to the **TC 2100 Group receptionist: 571-272-2100.**

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **Wei Y. Zhen** can be reached on **(571) 272-3708**. The fax phone number for the organization where this application or proceeding is assigned is **571-273-8300**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Satish S. Rampuria
Patent Examiner/Software Engineer
Art Unit 2191

Mary Steelman
Primary Examiner
2·22·2007